(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 694 830 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
31.01.1996 Bulletin 1996/05

(51) Int Cl.6: G06F 3/06

(21) Application number: 95106920.2

(22) Date of filing: 08.05.1995

(84) Designated Contracting States:
DE FR GB

(30) Priority: 26.07.1994 US 280419

(71) Applicant: International Business Machines Corporation
Armonk, N.Y. 10504 (US)

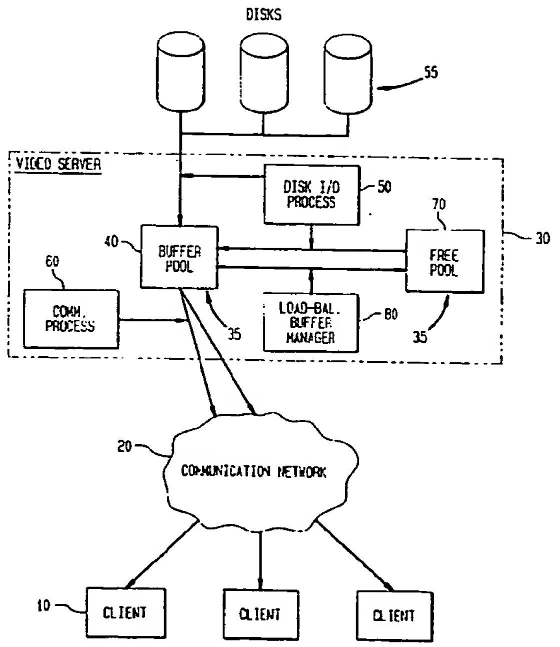
(72) Inventors:
- Dan, Asit
West Harrison, New York 10604 (US)
- Sitaram, Dinkar
Yorktown Heights, New York 10598 (US)

(74) Representative: Schäfer, Wolfgang, Dipl.-Ing.
D-70548 Stuttgart (DE)

(54) Buffering for load balancing in on-demand video servers

(57) A video server is provided with buffer manager which balances the loads on the various "movie storage" elements of a video server by preferentially buffering streams on highly loaded storage elements. The allocation of buffer takes place only when the storage element load increases due to the arrival of a new request or when buffer becomes available due to the pausing or stopping of an old request.

FIG. 1



EP 0 694 830 A1

Description

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of s/n 08/204,038 filed on March 1, 1994.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a movie (video) on demand systems of the type wherein multiple clients are serviced by video streams delivered from a central video server.

Related Art

With the advent of digital video technology, it is feasible to provide video-on-demand services to a large number of clients over a geographically distributed network. The videos are stored on secondary storage devices (such as disks and tapes) on the server and delivered to the clients. Because of non-uniform demand for different movies, load imbalances amongst the secondary storage devices may occur. In such situations, the most heavily loaded secondary storage device can become a bottleneck.

One solution to the load balancing problem is to statically replicate popular movies on multiple secondary storage devices based on the expected load, such that the total demand for the movie can be spread among the devices having a copy of the movie. This is, however, expensive in terms of the storage space required. Additionally, it is difficult to determine exactly how much replication is required for any particular movie since, typically, the demand can not be exactly forecast. Static replication is also not well suited to dealing with the problem of time-varying load since the number of replicas needed may vary.

Dynamic replication is more effective on load surges that occur on a time scale comparable to the time to copy a video file or segment. In dynamic replication, movies or portions of movies are copied, as a function of present demand, from heavily loaded storage devices to more lightly loaded storage devices in order to reduce the load. The effectiveness of dynamic replication in dealing with sudden load surges is limited by the bandwidth of the secondary storage devices.

SUMMARY OF THE INVENTION

In light of the above, the inventors have provided a buffer manager that balances the loads across the server disks by using information about load imbalances in order to decide which streams to buffer.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1	is a block diagram of a video-on-demand system according to an embodiment of the present invention;
FIG. 2	shows data structures maintained by the buffer manager of FIG. 1;
FIGs 3A - 3B	are a flow chart of the handling of a start or resume request by the buffer manager of FIG. 1;
FIGs. 4A - 4B	are a flow chart of moving a candidate stream by the buffer manager of FIG. 1;
FIG. 5	is a flow chart of the actions taken by the buffer manager of FIG. 1 on receipt of a stop request;
FIG. 6	is a flow chart of the steps taken by the buffer manager of FIG. 1 to find buffered streams when new buffer becomes available; and,
FIGs. 7A - 7B	show, respectively, the interactions between the disk I/O process and the communication process with the buffer manager of FIG. 1.
FIG. 8	is a block diagram of a system having a distribution node coupled to multiple server nodes by way of communication network.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

Overview

In accordance with an embodiment of the present invention, a buffer manager balances the loads on the various "movie storage" disks of a video server by preferentially buffering streams on highly loaded disks or other secondary storage devices. It should be understood, that while the present embodiment is described with respect to balancing the load across disks, the principles of the present invention are also applicable to balancing the load across other storage devices (such as tapes and jukeboxes) and across video servers. The allocation of buffer takes place only when the disk load increases due to the arrival of a new request or when buffer becomes available due to the pausing or stopping of an old request.

The amount of memory buffer required to enable a following stream to read the data from its preceding stream is a function of the time interval between the two streams and the compression method used. An estimate of the required buffer space can be obtained at the time when a new stream is created to form a new pair of consecutive streams.

The present buffer manager balances the load across the server disks by using information about load imbalances in order to decide which streams to buffer. At the time when a stream starts or restarts, it forms a consecutive pair with the stream for the same movie that is just ahead of it, if any. For any consecutive pair, the blocks brought in by the previous stream of the pair can be buffered so that the following stream reads them from buffer instead of disk. Therefore, the buffer requirement for the following stream of a consecutive pair is the amount of buffer required to store the required blocks.

For each disk, the buffer manager maintains a list of streams currently reading from that disk, in ascending order of buffer requirement. Note that if the movie is replicated, the preceding stream of the pair could be reading from a different disk than the succeeding stream in the pair.

One objective of the load balancing performed by the buffer manager is to reduce the load imbalance across disks. When a new stream creates a load imbalance on a disk, the buffer manager attempts to reduce the load of the highly loaded disk by switching a stream of that disk to the buffer. Load imbalance is said to exist if the difference between the load on the highly loaded disk and the average load per disk exceeds a threshold. The stream with the lowest buffer requirement and still being served from that disk is selected. If sufficient buffer is not available, the buffer manager attempts to buffer the selected stream from the highly loaded disks by stopping the buffering of streams from the lightly loaded disks until enough buffer is available.

The objective of the load balancing function of the buffer manager changes from load balancing to minimizing the loss if the total load on the server is greater than the total server disk bandwidth. In this case, the streams with the lowest buffer requirement across all disks for which the load exceeds their corresponding bandwidth capacity are chosen. Once the load on a particular disk falls below the maximum bandwidth of the disk, further streams on that disk are not considered for buffering.

The present buffer manager load balancing method can be used together with a method of dynamic replication to provide an integrated approach to dynamic replication and buffering in a video server. Buffering and dynamic replication algorithms have additional characteristics that are complementary and hence, make them suitable to different scenarios with respect to load surges.

Dynamic replication is most effective on load surges that occur on a time scale comparable to the time to copy a video file or segment. The present buffer manager, on the other hand, can handle sudden load surges. The integrated method thus can handle both sudden and slow load imbalances.

Detailed Description

Fig. 1 is a block diagram of a video-on-demand system according to an embodiment of the present invention. It is assumed that clients 10 make requests from a video server 30 via a communication network 20. Clients can submit start and stop requests. The communication network 20 can be, for example, and Asynchronous Transfer Mode (ATM) network comprising optical fiber cables.

The video server 30 is embodied within the framework of a general purpose computer such as an RISC SYSTEM/6000 workstation or an ES/9000 mainframe (both available from International Business Machines Corporation of Armonk, New York). As pertains to the embodiment of the present invention, the video server includes a buffer storage 35 which is instantiated in the video servers conventional semiconductor memory (not shown) and a number of processes 50, 60, and 80 which execute under control of the video servers operating system or main control program (not shown).

Movies (also referred to as video data) are stored in a number of disks 55 and/or other mass storage devices dedicated storing movies (generally referred to as secondary storage) which are connected to the video server. The disks 55 can be, for example, in the form of a DASD array or RAID system. Other disks (not shown) store working data and program code for the video server 30. The program code includes processes such as the servers main control program, a movie scheduling program, a customer usage tracking program, and the various communications, I/O and buffer management processes.

The buffer storage 35 is subdivided into two pools - a free pool 70 that contains free blocks that can be used for the storage of data and a buffer pool 40 that has blocks containing data ready for transmission to the users. The disk I/O process 50 reads video data from the disks 55 into empty buffers obtained from the free pool 70 and inserts the buffers into the buffer pool 40. The communication process 60 transmits video data from the buffer pool 40 or disks 55 to the clients 10. The load balancing buffer manager 80 allocates buffers to streams so as to reduce the load imbalance across the disks or minimize the number of streams being served from buffer depending of the current load on the system. Those skilled in the art will recognize that the disks 10 are coupled to the video server by way of a controller (not shown) which also includes its own internal memory buffer.

The video data that has been read into the buffer for a client can be retained for re-transmission to other clients. This conserves server resources since the retransmitted block does not have to be re-read from disk 55. Hence the buffer manager 80 selects a set of clients whose blocks will be retained in the buffer for subsequent re-transmission to other clients. Clients receiving re-transmitted blocks will be referred to as buffered clients.

A number of data structures are maintained by the buffer manager 80. These structures, shown in FIG. 2, include a stream table 110, a movie table 150, a disk table 175, a nominal free space counter 185 and an actual free space counter 190.

The stream table 110 contains one stream entry for each movie stream currently active. Each stream entry includes a number of fields 120-145. These fields contain, respectively, the identifier of the stream (Stream Id 120), the movie identifier (Movie Id 125), the identity of the disk (if any) the stream is reading from (Disk Id 127), the current block being displayed to the client or clients (current block 130), a read state (Read State 135) which identifies the source of the video data being provided to the clients, an indicator for the amount of buffer necessary to service the stream from the buffer pool (Buffer Reqd. 140), and a mark (Mark 145) indicating whether the source of the stream is moveable to a disk from the buffer.

The read state 135 can take on 4 values - DISK if the stream is reading from disk, BUFFER if the stream is a buffered stream, DISK_TO_BUFFER if the stream is in the process of being switched from the disk to the buffer and BUFFER_TO_DISK if the stream is in the process of being switched from the buffer to disk.

The mark field (mark 145) is used by the buffer manager when allocating buffer and can take on four values - blank, MOVEABLE if the stream can be switched from buffer to disk and UNMOVABLE if it cannot and CANDIDATE if the buffer manager is trying to switch the stream to the buffer.

The movie table 150 contains the movie id (Movie Id 155) and movie list for each active movie. The movie list is a list of stream elements (170) for that movie in ascending order of current block. The movie table includes a pointer to the first and last elements of the list (First str 160 and Last str. 165, respectively). Each stream element includes the identifier of the stream (Str. Id 167) and a pointer to the next stream element (Next 169).

The disk table 175 keeps track of the load on the disks 55. For each disk, the disk table contains the identifier of the disk (disk id 178) the actual load on the disk (180) and the potential load (182). The potential load represents the number of streams that can be switched to this disk in order to free buffer for reducing the load on another disk. It is used by the buffer manager 80 when allocating buffer to the disk.

The two counters - the nominal free space 185 and the actual free space 190 are used to keep track of the amount of buffer space available in the free pool 70. The actual free space counter 190 is initialized to the total amount of buffer available. The actual free space counter is used to keep track of the actual amount of buffer available in the free pool 70.

The nominal free space counter is initialized to some high fraction (such as 90%) of the total buffer space available. The nominal free space counter is used by the buffer manager 80 when allocating and deallocating space; i.e. when allocating buffer space, the buffer manager allocates only the amount of space specified by the nominal free space counter. This is because fluctuations in compression ratios may cause the actual amount of buffer required by a stream to vary. Hence, at times a stream may require more or less buffer space than allocated to it by the buffer manager. The difference between the initial values of the nominal and actual free space counters (10% in the example) is a safety buffer used for accommodating this variation.

A flow chart of the handling of a start request by the buffer manager is shown in Figs. 3A-3B After a new request arrives in step 210, in step 220 the buffer manager creates a new stream entry for the request and inserts the new stream entry into the stream table 110. The read state of the stream entry is set to DISK. The current block 130 is set to the requested block and the stream entry is inserted into the movie list for the movie using the current block to determine its position.

In step 230, the buffer manager scans the movie list 170 to determine the closest preceding stream if any for the movie; i.e. the stream among all the streams reading this movie whose current block is closest to the current block of this request. If no such block exists, in step 240 buffer manager exits the new request handling process. If such a request exists, in step 250 the buffer manager sets the buffer reqd. field 140 in the stream entry to the amount of storage required to buffer all the blocks between the immediately preceding stream and the current request. In step 255 the buffer manager 80 determines the buffering mode. If the loads on all disks are greater than their corresponding load threshold, the buffer manager switches to the mode of maximizing the number of streams being served from the buffer. This can be accomplished as shown, for example in United States Patent Application s/n 08/204,038 filed on March 1, 1994, which is incorporated by reference herein as if printed in full below. Otherwise, the buffer manager selects a disk (from among the disks 55) from which to service this request and updates the load on the selected disk in step 260.

Next, in step 270, the buffer manager determines if the load on the selected disk is above a pre-specified threshold. If not, in step 280 buffer manager exits the new request handling process. If the load is above the threshold, in step 290-the buffer manager examines the stream table 110 to find the stream that is being serviced from the disk (read state 135 equal to BUFFER or BUFFER_TO_DISK) that has the lowest buffer requirement 140 and sets the mark field 145 of the stream entry for that particular stream to CANDIDATE. Then, in step 295 the buffer manager invokes the "Move candidate stream" procedure of Figs. 4A and 4B and exits the new request handling process.

Figs. 4A and 4B describe the buffer manager's "Move candidate stream" procedure. In step 300 the buffer manager examines the available space to determine if enough free buffer space exists to accommodate the candidate request. The available space is the nominal free space 185 together with the space held by moveable streams (streams with mark 145 equal to MOVEABLE). In the first pass through the loop, there will not be any such streams. If enough buffer space is available, the buffer manager initiates steps 370-410 to buffer the request.

If enough free buffer is not available, the buffer manager attempts to free some space by displacing some other stream. First, in step 310, the buffer manager checks if there are any more unmarked streams. If there are none, no additional buffer can be freed, hence the buffer manager unmarks all the stream entries and resets the disk potential loads in step 315 and, in step 320 returns failure to the invoking process. If there are unmarked streams, in step 330 the buffer manager locates the unmarked stream with the largest buffer reqd 140. Then, in step 340, the buffer manager determines whether this stream can be moved to the disk and its buffer freed. The unmarked stream can be moved if moving the stream to disk does not cause the projected load on the unmarked stream's disk to exceed the projected load on the candidate stream's disk.

The projected load on the unmarked stream's disk is the sum of the actual load 180, the potential load 182 and the load imposed by the unmarked stream. The projected load on the candidate stream's disk is the actual load 180 minus the load of the candidate stream. If the unmarked stream cannot be moved, in step 360 its mark field 145 is set to UNMOVABLE. If it can be moved, in step 350 its mark field 145 is set to MOVEABLE and the potential load field 182 of the unmarked stream's disk is incremented by the load of the unmarked stream. In either case, the buffer manager then loops back to step 300.

If, in step 300, it is determined that the available space is greater than the buffer requirement of the candidate stream (i.e. the candidate stream can be moved), the buffer manager executes step 370. In step 370, the buffer of all streams marked moveable are released by incrementing the nominal free space counter 185 by buffer reqd field 140 of the stream. Then, in step 380, the sources of the streams are moved to disks by setting the read state 135 of the streams to DISK. In step 390, the nominal free space counter 185 is decremented by the buffer reqd. field 140 of the candidate stream and the read state 135 of the candidate streams is set to DISK_TO_BUFFER, effectively allocating buffer to the candidate stream. Finally, in step 400, the actual load 180 of the disks are updated to reflect the new loads on the disks and the potential load fields 182 of the disks are reset to 0 and the mark field 145 of the is set to blank. The buffer manager then returns success the invoking process in step 410.

The actions taken by the buffer manager on receipt of a "stop" request from a client are shown in Fig. 5. A stop request is a request from a client to terminate showing of the movie. In step 520, the stream entry for the stream is removed from the stream table 110 and the movie list in the movie table 150 In step 530, the buffer manager determines if the stream had allocated buffer by examining its read state 135 (BUFFER or DISK_TO_BUFFER). If not, the buffer manager exits the stop request handling procedure in step 540. If the stream had allocated buffer, in step 550 the buffer manager frees the buffer by incrementing the nominal free space counter 185 by the buffer reqd. 140. In step 555 the buffer manager determines the buffering mode (either load balancing or maximizing the number of streams served from the buffer). If the loads on all disks are still greater than their corresponding load threshold, then buffer manager goes into (or stays in, as the case may be) the mode of maximizing the number of streams being served from the buffer. Otherwise, the buffer manager then executes the "Look for new buffered stream" steps shown in Fig. 6. It should be understood that the principles of present load balancing buffer manager can also be applied to client "pause" requests.

Fig. 6 shows the steps taken by the buffer manager to find non-buffered streams to buffer when new buffer becomes available. In step 610, the buffer manager makes a list of disks whose actual load is above a predetermined load imbalance threshold. The load imbalance threshold can be specified by the system administrator or determined by a load balancing process executing on the system. Then, in step 620, the buffer manager starts a loop over the disks in the list in descending order of load. For the selected disk, the buffer manager finds the stream with the lowest buffer requirement and marks this stream (mark field 145) as a candidate stream. It then executes the "Move candidate stream" procedure (of FIG. 4) in step 640. When the move candidate procedure has been completed, in step 650 the buffer manager then checks to see if the loop is complete and if so, exits the "Look for new stream to buffer" procedure in step 660. Otherwise, the buffer manager selects the next disk in step 670 and then re- executes the loop starting again at step 630.

Figs 7A and 7B shows the interactions between the disk I/O process 50, the communication process 60 and the load balancing buffer manager 80. In step 710, the disk I/O process 50 receives a request from a client to read a block from the disk. Before initiating an I/O operation to read a requested block, in step 720 the disk I/O process 50 looks in the buffer pool in step 720 to determine if the requested block has been read in. If it has, in step 723 the disk I/O process checks the read state of the stream to see if it is DISK_TO_BUFFER. If it is not, the disk I/O process exits in step 730. If it is DISK_TO_BUFFER, in step 726 the disk I/O process changes the read state to BUFFER and then exits in step 730. If not, in step 740 the disk I/O process gets a free buffer from the free pool, decrements the actual free space counter starts the I/O and then exits in step 750.

After transmitting a block, the communication process 60 checks the read state of the following stream in step 760. If the read state is BUFFER or DISK_TO_BUFFER, the communications process exits in step 770. Otherwise, in step 765 the communication process increments the actual free space counter 190 and returns the block to the free pool 70.

FIG. 8 is a block diagram of a system having a distribution node coupled to multiple server nodes by way of communication network. In this embodiment the distribution node includes the load balancing buffer manager which balances the load across the video servers in accordance with the foregoing principles.

Now that the invention has been described by way of the preferred embodiment, various modifications and improvements will occur to those of skill in the art. Thus it should be understood that the preferred embodiment has been provided as an example and not as a limitation. The scope of the invention is defined by the appended claims.

Claims

1. A method of managing memory buffer in a video server, wherein a plurality of clients are served from video streams provided from disks, comprising the steps of:
determining buffer requirements of a plurality of the video streams, the buffer requirements being a number of frames separating each video stream from an immediately previous video stream carrying the same video;
creating a list of streams being served from each of-the disks, ordered by the buffer requirement of each of the streams;
determining when to balance the load across the disks;
allocating the buffer to the streams on a most heavily loaded one of the disks so as to serve the streams from the buffer, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied; and
retaining blocks of the immediately preceding stream in the buffer allocated to its following stream and discarding the blocks from the buffer as they are read by a client viewing the following stream.

2. The method of Claim 1 comprising the further steps of:
updating the list when any of starting, stopping pausing and resuming of a video occurs; and,
responsive to the updating, identifying the most heavily loaded one of the disks after the updating and reallocating the buffer to the streams being served from the most heavily loaded disk, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied.

3. A method of managing memory buffer in a video server, wherein a plurality of clients are served from video streams provided from disks, comprising the step of:
determining buffer requirements of a plurality of the video streams, the buffer requirements being a number of frames separating each video stream from an immediately previous video stream carrying the same video;
creating a list of streams being served from each of the disks, ordered by the buffer requirement of each of the streams;
selecting a buffering mode from one of (a) balancing the load across the disks and (b) maximizing streams that can be served from the buffer;
when the mode is balancing the load across the disks, allocating the buffer to the streams on a most heavily loaded one of the disks so as to serve the streams from the buffer, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied; and,
when the mode is maximizing the streams that can be served from the buffer: examining the buffer requirements and as a function thereof, allocating the buffer so as to maximize a number of streams that can be provided therefrom regardless of the load distribution across the disks.

4. A system for managing memory buffer in a video server wherein a plurality of clients are served from video streams provided from disks, comprising:
means for determining buffer requirements of a plurality of the video streams, the buffer requirements being a number of frames separating each video stream from an immediately previous video stream carrying the same video;
means for creating a list of streams being served from each of the disks, ordered by the buffer requirement of each of the streams;
means for determining when to balance the load across the disks;
means for allocating the buffer to the streams on a most heavily loaded one of the disks so as to serve the streams from the buffer, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied; and

means for retaining blocks of the immediately preceding stream in the buffer allocated to its following stream and discarding the blocks from the buffer as they are read by a client viewing the following stream.

5. A method of managing memory buffer in a video server, wherein a plurality of clients are served from video streams provided by video storage elements, comprising the steps of:
determining buffer requirements of a plurality of the video streams, the buffer requirements being a number of frames separating each video stream from an immediately previous video stream carrying the same video;
creating a list of streams being served from each of the storage elements, ordered by the buffer requirement of each of the streams;
allocating the buffer to the streams on a most heavily loaded one of the storage elements so as to serve the streams from the buffer, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied; and,
retaining blocks of the immediately preceding stream in the buffer allocated to its following stream and discarding the blocks from the buffer as they are read by a client viewing the following stream.

6. The method of Claim 6 comprising the further steps of:
updating the list when any of starting, stopping, pausing and resuming of a video occurs; and,
responsive to the updating, identifying the most heavily loaded one of the disks after the updating and reallocating the buffer to the streams being served from the most heavily loaded disk, starting from a stream with a smallest buffer requirement and proceeding to streams with larger buffer requirements until the buffer requirement of a stream can not be satisfied.

FIG. 1

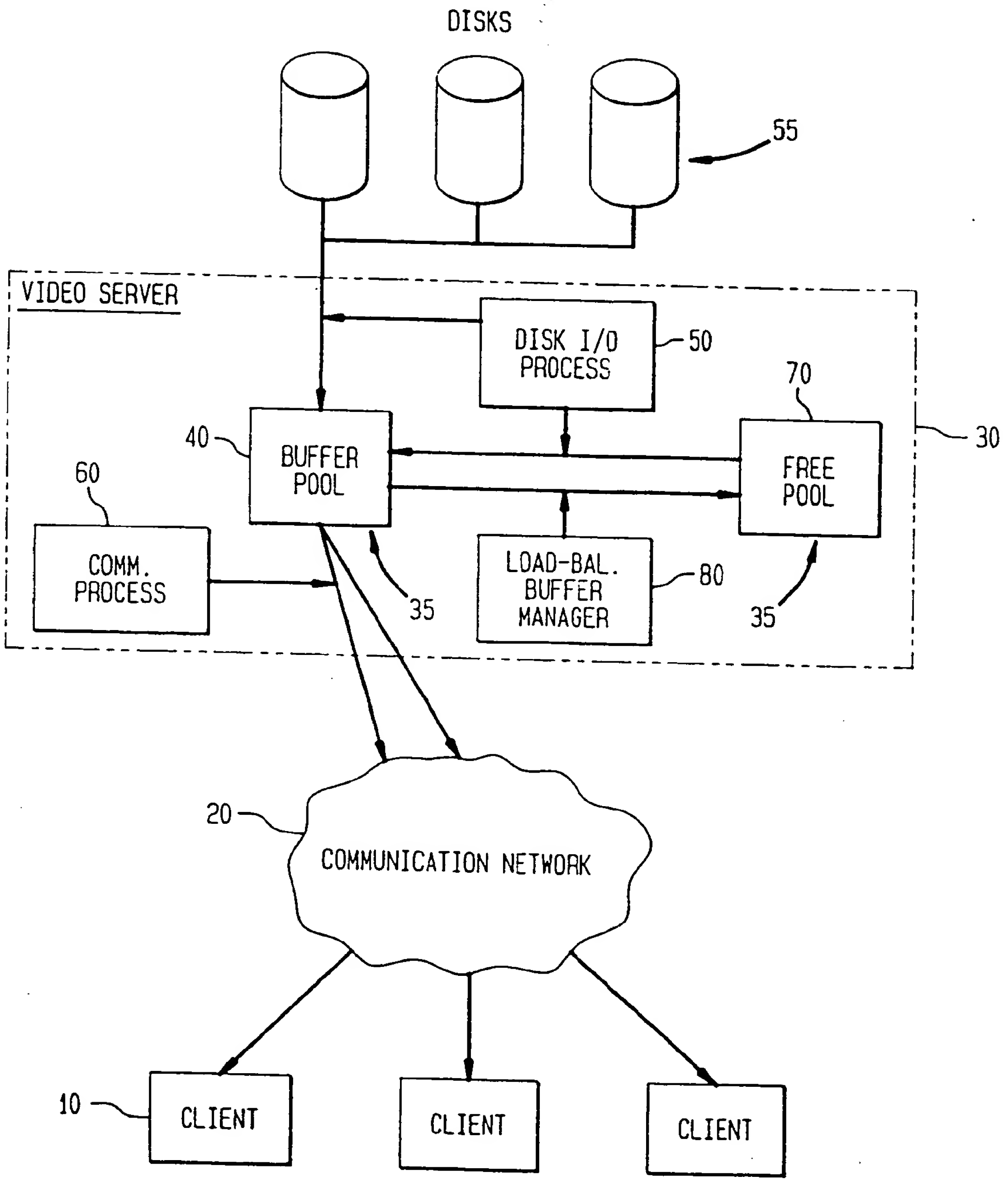
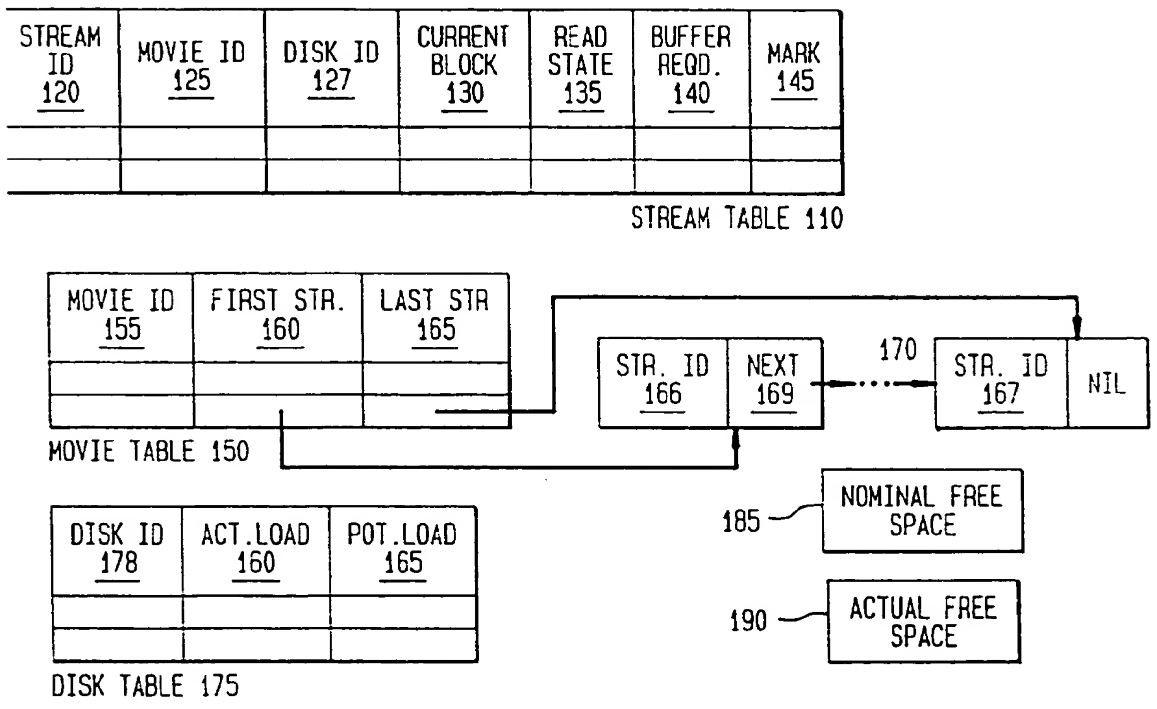

DISKS
55
VIDEO SERVER
DISK I/O PROCESS
50
70
30
40
BUFFER POOL
FREE POOL
60
COMM. PROCESS
LOAD-BAL. BUFFER MANAGER
80
35
35
20
COMMUNICATION NETWORK
10
CLIENT
CLIENT
CLIENT

FIG. 2



STREAM ID 120
MOVIE ID 125
DISK ID 127
CURRENT BLOCK 130
READ STATE 135
BUFFER REQD. 140
MARK 145
STREAM TABLE 110
MOVIE ID 155
FIRST STR. 160
LAST STR 165
MOVIE TABLE 150
STR. ID 166
NEXT 169
170
STR. ID 167
NIL
DISK ID 178
ACT.LOAD 160
POT.LOAD 165
DISK TABLE 175
185
NOMINAL FREE SPACE
190
ACTUAL FREE SPACE

FIG. 3

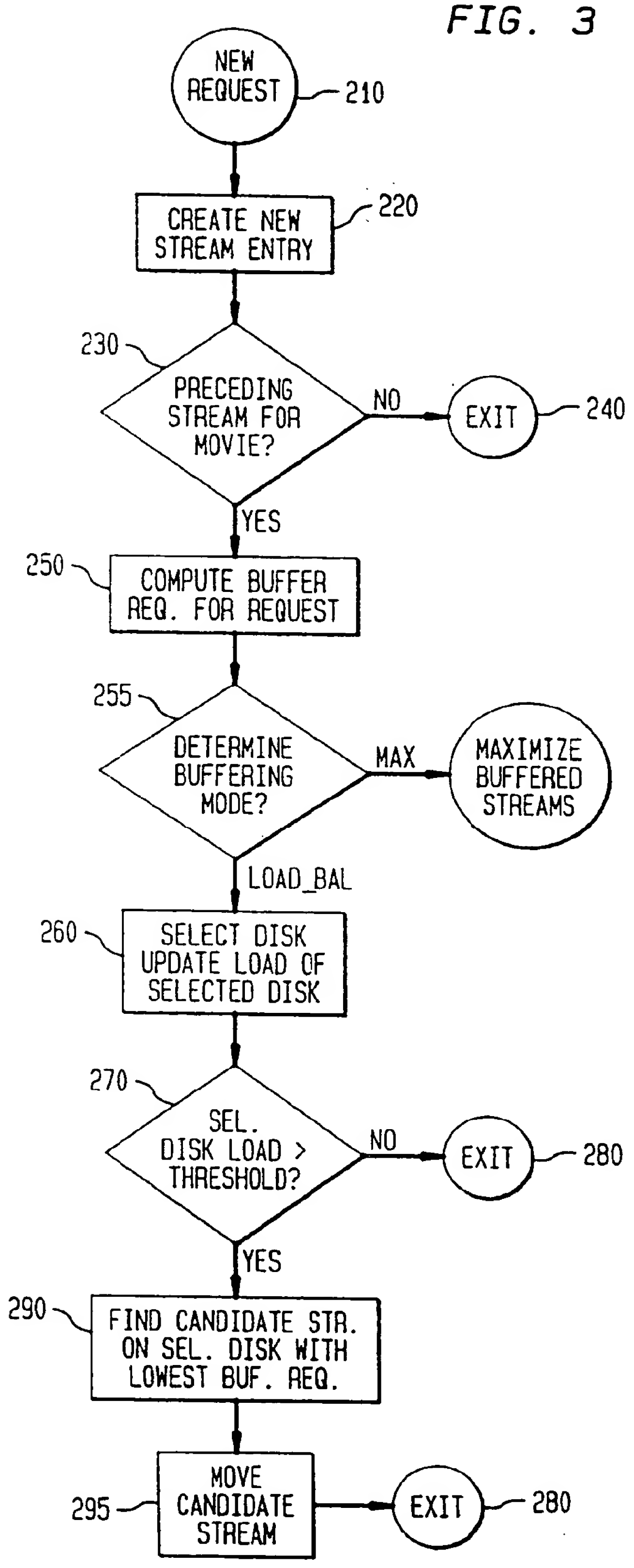

NEW REQUEST
210
CREATE NEW STREAM ENTRY
220
230
PRECEDING STREAM FOR MOVIE?
NO
EXIT
240
YES
250
COMPUTE BUFFER REQ. FOR REQUEST
255
DETERMINE BUFFERING MODE?
MAX
MAXIMIZE BUFFERED STREAMS
LOAD_BAL
260
SELECT DISK UPDATE LOAD OF SELECTED DISK
270
SEL. DISK LOAD > THRESHOLD?
NO
EXIT
280
YES
290
FIND CANDIDATE STR. ON SEL. DISK WITH LOWEST BUF. REQ.
295
MOVE CANDIDATE STREAM
EXIT
280

FIG. 4

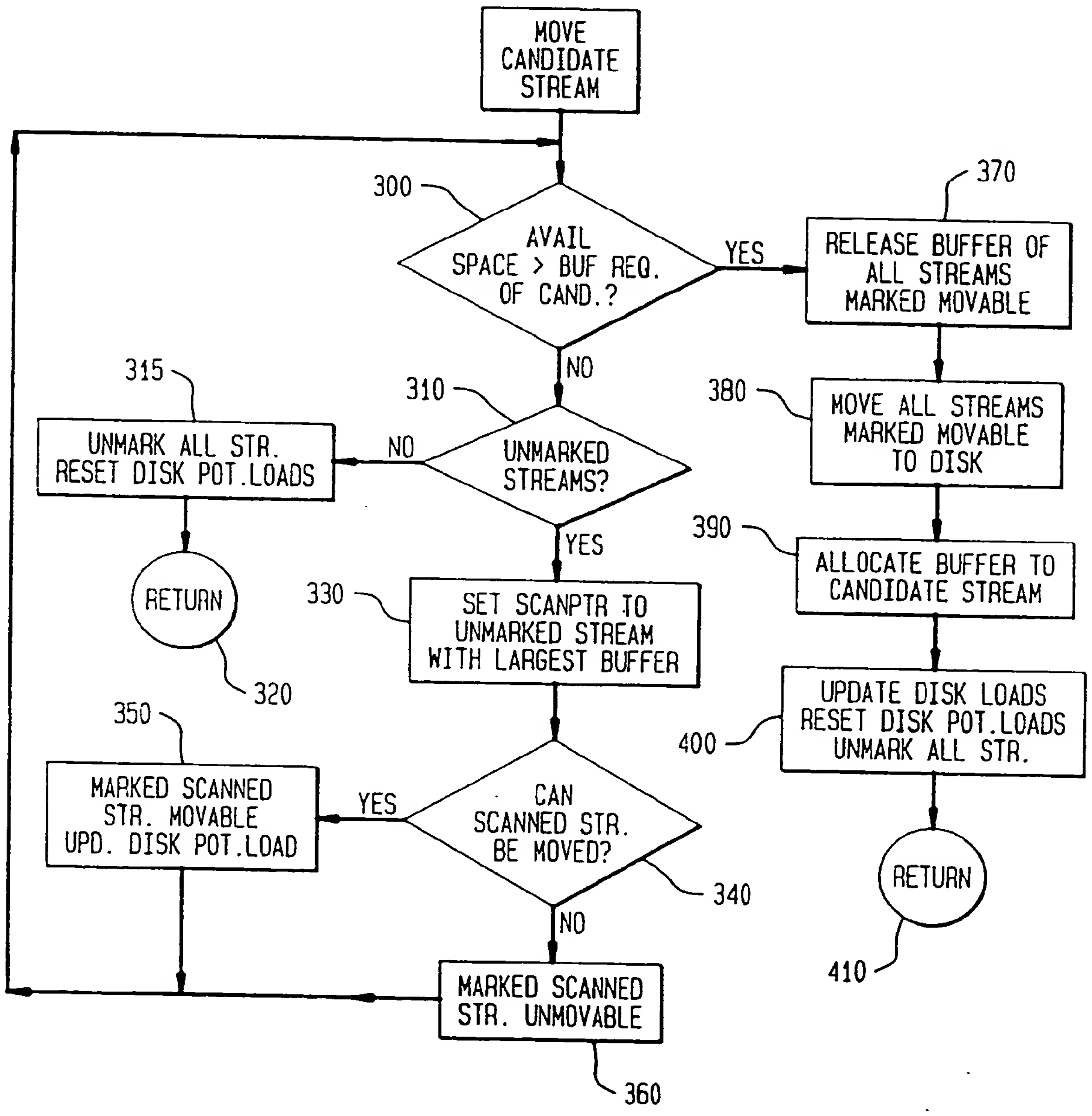

MOVE CANDIDATE STREAM
300
AVAIL SPACE > BUF REQ. OF CAND.?
YES
NO
370
RELEASE BUFFER OF ALL STREAMS MARKED MOVABLE
380
MOVE ALL STREAMS MARKED MOVABLE TO DISK
390
ALLOCATE BUFFER TO CANDIDATE STREAM
400
UPDATE DISK LOADS RESET DISK POT.LOADS UNMARK ALL STR.
RETURN
410
310
UNMARKED STREAMS?
NO
YES
315
UNMARK ALL STR. RESET DISK POT.LOADS
RETURN
320
330
SET SCANPTR TO UNMARKED STREAM WITH LARGEST BUFFER
340
CAN SCANNED STR. BE MOVED?
YES
NO
350
MARKED SCANNED STR. MOVABLE UPD. DISK POT.LOAD
360
MARKED SCANNED STR. UNMOVABLE

FIG. 5

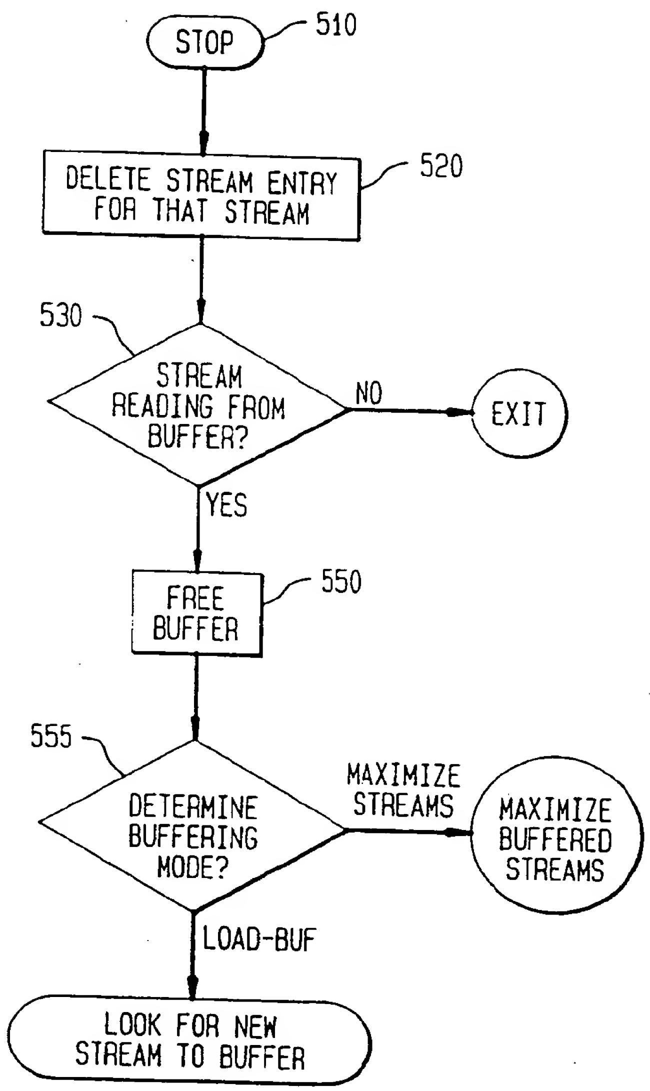

STOP
510
DELETE STREAM ENTRY FOR THAT STREAM
520
530
STREAM READING FROM BUFFER?
NO
EXIT
YES
FREE BUFFER
550
555
DETERMINE BUFFERING MODE?
MAXIMIZE STREAMS
MAXIMIZE BUFFERED STREAMS
LOAD-BUF
LOOK FOR NEW STREAM TO BUFFER

FIG. 6

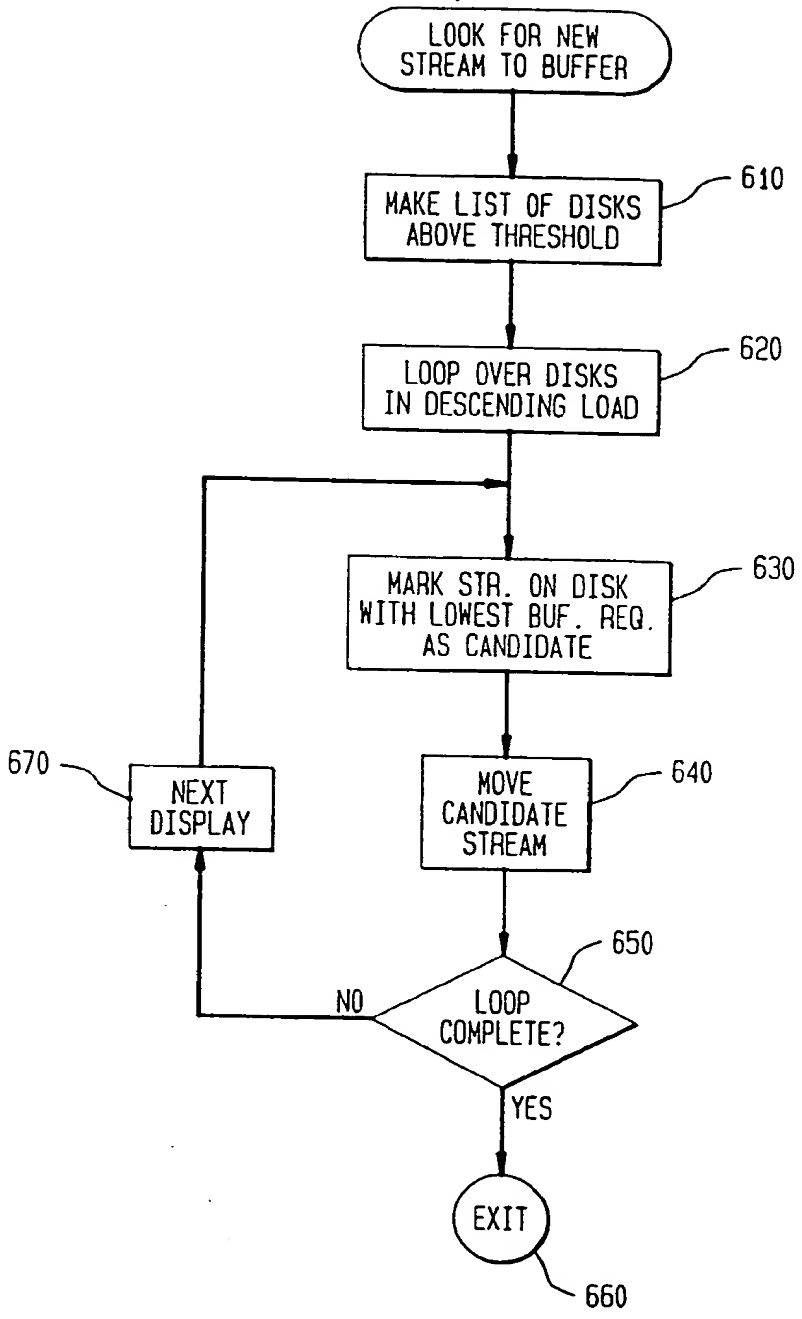

LOOK FOR NEW STREAM TO BUFFER
MAKE LIST OF DISKS ABOVE THRESHOLD
610
LOOP OVER DISKS IN DESCENDING LOAD
620
MARK STR. ON DISK WITH LOWEST BUF. REQ. AS CANDIDATE
630
670
NEXT DISPLAY
MOVE CANDIDATE STREAM
640
650
NO
LOOP COMPLETE?
YES
EXIT
660

FIG. 7A

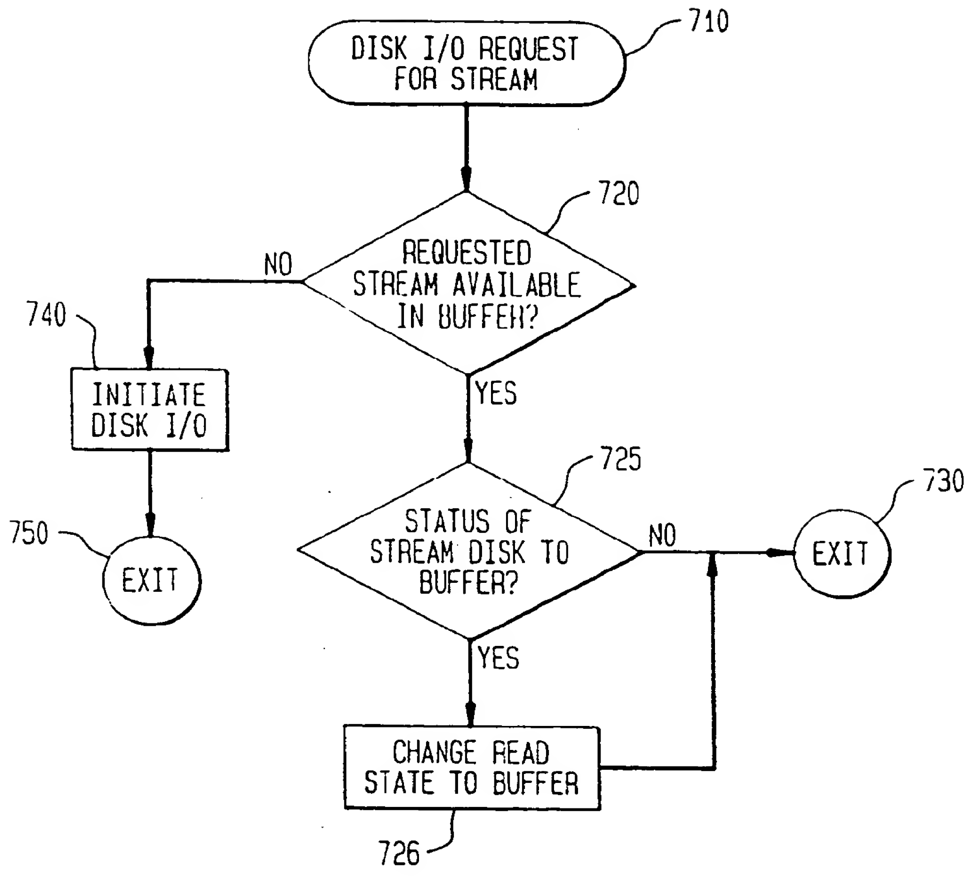

710
DISK I/O REQUEST FOR STREAM
720
REQUESTED STREAM AVAILABLE IN BUFFER?
NO
YES
740
INITIATE DISK I/O
750
EXIT
725
STATUS OF STREAM DISK TO BUFFER?
NO
YES
730
EXIT
CHANGE READ STATE TO BUFFER
726

FIG. 7B

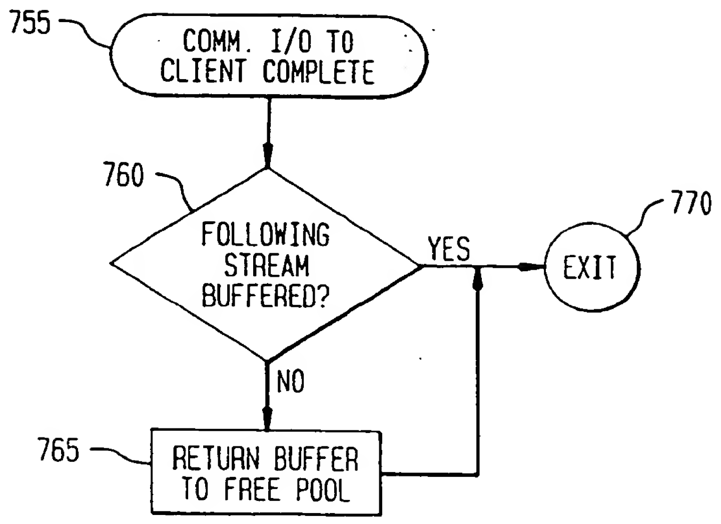
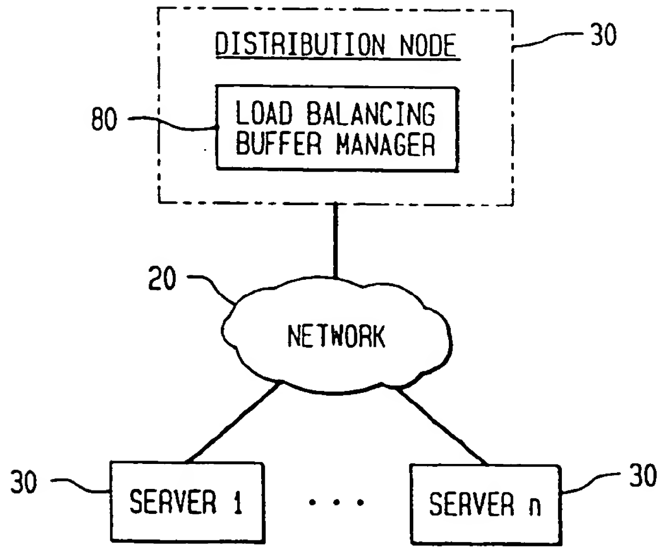

755
COMM. I/O TO CLIENT COMPLETE
760
FOLLOWING STREAM BUFFERED?
YES
NO
770
EXIT
765
RETURN BUFFER TO FREE POOL

FIG. 8


DISTRIBUTION NODE
30
80
LOAD BALANCING BUFFER MANAGER
20
NETWORK
30
SERVER 1
. . .
SERVER n
30

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 10 6920

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	PROC. SPIE STORAGE AND RETRIEVAL FOR IMAGE AND VIDEO DATABASES II, 7 February 1994, SAN JOSE, CALIFORNIA, USA pages 208 - 221 ED CHANG ET AL. 'Scalable Video Data Placement on Parallel Disk Arrays' * page 209, line 1 - line 16; figure 1 * ---	1	G06F3/06
A	1994 IEEE 13TH ANNUAL CONF. ON COMPUTERS AND COMMUNICATIONS, 12 April 1994, PHEONIX, USA pages 440 - 447, XP000462593 KINGSLEY C. NWOSU 'A Buffer Management Strategy for a Model of Digital Continuous Media Display' -----		
			TECHNICAL FIELDS SEARCHED (Int.Cl.6)
			G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	18 August 1995	Absalom, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)